

适用于动态软件体系结构的扩展的Z描述语言*

周绪川

(西南民族大学 计算机科学与技术学院, 成都 610041)

摘要: 动态软件体系结构语言已成为描述复杂软件体系结构的重要工具,然而许多描述语言都是静态的,并不能对动态软件体系结构进行描述。为此,对经典Z描述语言进行扩展,主要通过对构件、连接件和它们的添加以及删除来达到动态演化的目的。实例分析表明了这种扩展的可行性。

关键词: 软件工程; 动态演化; 动态软件体系结构; Z语言描述

中图分类号: TP301 **文献标志码:** A **文章编号:** 1001-3695(2012)09-3338-03

doi:10.3969/j.issn.1001-3695.2012.09.036

Extended Z description language for dynamic software architecture

ZHOU Xu-chuan

(College of Computer Science & Technology, Southwest University for Nationalities, Chengdu 610041, China)

Abstract: Dynamic software architecture language has become an important tool which describes complex software architecture. However, many description languages are static and can't describe the dynamic software architecture. So, this paper extended the classical Z description language that described dynamic evolution by means of components, connectors and their connection which added and deleted. Example analysis shows that the extended Z description language is feasible.

Key words: software engineering; dynamic evolution; dynamic software architecture(DSA); Z description language

0 引言

软件体系结构(SA)指的是从全局视角描述系统的构件组成、构件之间的相互关系和连接方式所形成的拓扑结构及其所遵循的模式和受到的约束等^[1]。动态软件体系(DSA)则是指那些在运行时刻会发生变化的体系结构^[2]。软件体系结构语言(ADL)使用符号标记把软件体系结构分解成构件和连接件,并说明这些元素如何连接在一起构成一个配置^[3]。它不但是形式化描述软件体系结构的基本工具,而且还是对软件体系结构进行求精、验证、演化和分析的前提和基础。

目前,软件体系结构语言种类繁多,包括 ACME、AESOP、AML、ARMANI、CHAM-ADL、C2、Rapide 等^[4]。然而,这些软件体系结构语言多是静态的,并不能对动态软件体系进行描述。虽然也有些软件体系结构语言能够对体系结构进行动态描述,如 DARWIN、Dynamic-ACME、dynamic-WRIGHT 和 PAPIDE,但是它们这种支持都是有限的^[5]。例如,C2 是一种基于构件和消息的体系结构风格,为体系结构演化提供了特别的支持。但 C2 没有严密的形式化基础,不能对体系的动态行为进行严格的分析和推演;Rapide 不允许单独对连接件进行描述和分析,并且没有提供相关机制将多个连接机制捆绑成一个整体,构成复杂的交互模式。

Z 语言^[6]是一种基于集合理论和一阶谓词逻辑的形式语言或方法,它支持软件的形式化规格、规格的推理及求精,是迄今为止应用最为广泛的形式语言之一,但是也不支持动态软件体系的描述。为此,本文对 Z 语言进行扩展,以使之应用于动态软件体系描述之中。

1 动态软件体系结构介绍

1.1 动态特性

SA 通常是对系统的静态描述,如果需要改变体系结构则必须重新设计新的 SA,这已不能适应现在越来越多的需要在运行时刻发生变化的系统的设计需求;而 DSA 则允许系统在执行过程中修改其体系结构。体系结构的动态变化可分为如下几个方面^[7]:

a) 结构方面。软件系统为适应当前的计算环境,往往需要调整自身的结构,如增加或删除构件、连接子,这将导致 SA 的拓扑结构发生显式的变化。

b) 行为方面。由于用户需求的变化或者系统自身 QoS 调节的需要,软件在运行过程中会改变其行为。

c) 属性方面。已有的 ADL 大多支持对非功能属性(non functional properties)的规约和分析,在系统运行的过程中这些要求可能发生改变,而这些变化又会进一步触发软件系统结构或行为的调整。属性的变化是驱动系统演化的主要原因。

d) 风格方面。软件体系结构风格代表了相似的软件系统的基本结构以及相关的构造方法。

1.2 体系结构

体系结构一般是指系统的组织结构、它们之间的关联关系以及支配系统设计的原则和方针。一个体系结构的软件结构包括构成系统计算单元的构件、规范构件间交互行为的连接件以及构件和连接件如何组织在一起的配置。

构件(component)是体系结构的基本要素之一,构件是指具有一定功能、可明确辨识的软件单位,并具有以下特性^[8]:

a)独立部署单元;b)作为第三方的组装单元;c)没有(外部)可见状态。

一个构件可以独立部署,意味着它必能与它所在的环境及其他构件完全分离。如果第三方厂商能将一个构件和构件组装在一起,具有良好的内聚性,能够封装它的行为,并且只通过定义的接口与外部环境进行交互。

连接件(connector)^[6]是用来建立构件间的交互以及支配这些交互规则的体系结构构造模块。构件之间交互包括消息或信号量的传递、功能或方法调用、数据的传送和转换、构件之间的同步关系、依赖关系等,而这些都是通过连接件来实现。

体系结构配置(configuration)确定体系结构的构件与连接件连接关系的拓扑要求。体系结构配置提供以确定构件是否正确连接、接口是否匹配、连接件的通信是否正确,并说明实现要求行为的组合语义。

2 Z语言描述的扩展

2.1 Z语言描述

Z语言^[6]是一种基于集合理论和一阶谓词逻辑的形式语言或方法。Z支持软件的形式化规格、规格的推理及求精,是迄今为止应用最为广泛的形式语言之一。

数据抽象和过程抽象是软件规格过程中两类重要抽象。数据抽象是利用抽象的数据结构(如关系、函数等)来进行功能性的描述,而不关心这些抽象数据结构在计算机中是如何表示和实现的;过程抽象是指忽略任务具体完成的过程,而只精确描述该任务所要完成的功能,即描述了从输入到输出的映射,该映射的定义域和值域均使用数据抽象来刻画。

Z支持数据抽象和过程抽象,并表示为表示抽象一操作。在表示抽象中,数据从数据结构的表示细节抽象出来,使用关系、函数、集合、序列、包等;而操作则描述了在数据抽象中所引入的数据上的抽象算法与操作。

模式是Z语言的基本描述单位,一个软件系统的Z主要是由若干个模式构成,这些模式刻画了系统的静态性质和动态行为。一个模式有一个唯一的名字,并包括一个声明部分和一个断言部分或谓词部分。模式的声明部分引入了某些类型的变量,这些变量为模式内的局部变量;断言部分描述了在这些局部变量之间,或者局部变量与在该模式之前声明的全局变量之间的不变式关系。可以简单表示为如图1所示的形式。

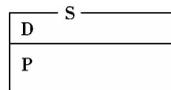


图1 Z语言模式

其中:S表示模式的名称;D表示声明部分;P表示断言部分或者谓词部分。

2.2 扩展后Z语言的描述规则

采用扩展的Z语言来描述软件体系结构的动态特性,其描述规则如下:

规则1 构件可以表示一个数据类型,接口同样可以表示一个数据类型。同样,连接件和接口也可以表示数据类型。

规则2 接口是用来表示接收还是发出请求的,其接口应该是属于{receive, send}两种类型,定义了接口的具体行为。

规则3 模式名字可以定义一个具体的接口、构件或系

统,其模式可包括其他的模式来表示其结构和行为。

规则4 构件之间的连接是通过连接件来实现的,其连接行为也可以定义是一个类型。

扩展的Z语言从语义与语法上描述了体系结构中构件、连接件和配置,如图2所示。

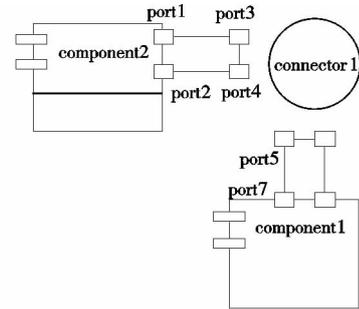


图2 体系结构中构件、连接件和配置情况

2.3 扩展后Z语言对构件的描述

构件是描述本地化、独立的计算。如图2在Z-ADL中对构件的描述包括两个方面,即接口(interface)和计算(computing)。接口是由多个端口(port)组成,每个端口表示构件参与的一种交互。对构件计算部分的描述说明了构件行为。如图2中的component1和component2,这两个构件通过连接件connector1来连接,从而实现了两个构件的相互通信。

扩展后Z语言对构件的描述如下:

```

component:
port1:port
port2:port
port1 ∈ receive
port2 ∈ send
#(port1) < 3 ∧ #(port1) < 8
#(port2) < = 5 ∧ #(port2) > = 1

```

其中:#(port)表示端口接收或发出数据长度。

2.4 扩展后Z语言对连接件的描述

连接件是一种特殊的构件,旨在建立构件间的交互以及支配这些交互规则。连接件依据接口和路由行为描述,显然连接件的接口也是由一组端口构成。

扩展后Z语言对连接件的描述如下:

```

connector:
port1:port
port2:port
port3:port
port4:port
port1, port2 ∈ receive
port3, port4 ∈ send
#(port1) < 3 ∧ #(port1) > 8
#(port2) < = 5 ∧ #(port2) > = 10

```

2.5 扩展后Z语言对配置的描述

配置是实现构件与连接件之间的连接从而形成一个完整的系统,其扩展后Z语言对配置的描述如下:

```

system:
component1:component
component2:component
connector1:connector
port1, port2, port3, port4:port

```

```

port5, port6, port7, port8:port
component1 to component2 :component  $\wedge$  connector  $\wedge$  component
port1, port2:component1
port3, port4, port5, Port6:connector1
port7, port8:component2
component1 to component2 =
component1. port1  $\wedge$  connector1. port3  $\wedge$ 
component1. port2  $\wedge$  connector1. port4  $\wedge$ 
connector1. port5  $\wedge$  component2. port7  $\wedge$ 
connector1. port6  $\wedge$  component2. port8
#( port1 ) < 3  $\wedge$  #( port1 ) > 8
#( port2 ) < = 5  $\wedge$  #( port2 ) > = 10
port1, port4, port6, port7  $\in$  receive
port2, port3, port5, port8  $\in$  send

```

其中： \wedge 表示连接关系； $R \wedge Q$ 表示 R 与 Q 相连，可表示一种数据类型。

3 动态演化实例分析

动态演化是动态软件体系的标志性之一，实验中采用扩展的 Z 语言来描述一个应用系统：学生管理系统。

该学生管理系统由两个构件构成，即学生注册系统 (SR) 和学生信息系统 (SI)，具体如图 3 所示。其中构件 SR 与构件 SI 是通过 SRI 连接件连接起来。

采用扩展的 Z 语言对 SR 和 SI 描述如下：

```

SCS:
SR:component
SI:component
SRI:connector
Req, Con, In, Out:port
In1, Out1, Rep, Com:port
SR to SI:SR  $\wedge$  SRI  $\wedge$  SI
Req, Con  $\in$  SR
In, Out, In1, Out1  $\in$  SI
Com, Rep  $\in$  SI
SR to SI = SR. Req  $\wedge$  SRI. In  $\wedge$  SI  $\wedge$  SR. Con  $\wedge$  SRI. Out  $\wedge$ 
SRI. In1  $\wedge$  SI. Com  $\wedge$  SI  $\wedge$  SRI. Out1  $\wedge$  SI. Req
Con, In, In1, Rep  $\in$  receive
Req, Out, Out1, Com  $\in$  send
#( In ) < 3  $\wedge$  #( In ) > 8
#( In1 ) < = 5  $\wedge$  #( In1 ) > = 10
#( Con ) < 3  $\wedge$  #( Con ) > 8
#( Com ) < = 5  $\wedge$  #( Com ) > = 10

```

对于一些新的用户来说，要进行注册才能成功访问信息，这样就产生了一个新构件为 UNS 代表了新用户，通过将 UNS 添加到学生管理系统中去，从而形成一个动态演化的系统，如图 3 所示。

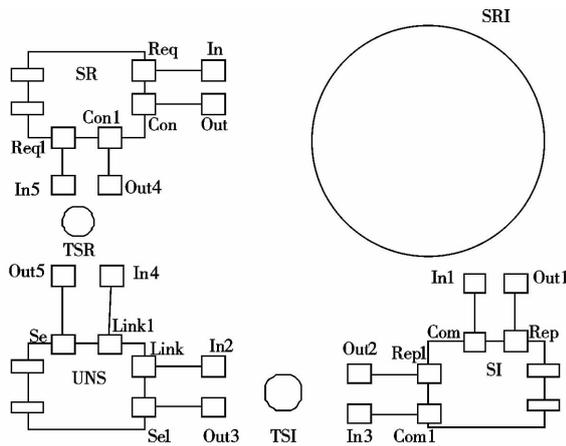


图3 动态演化系统

用扩展的 Z 语言对 UNS 构件添加到 SR 与 SI 连接的系统的描述如下：

```

SCSAdd:
SCS
UNS:component
TSR,TSI:connector
Con1,Req1,Se,Se1,Link,Link1.Req1,Com1:port
In2,In3,In4,In5,Out2,Out3,Out4,Out5:port
UNS to SR:SR  $\wedge$  TSR  $\wedge$  UNS
UNS to SI:UNS  $\wedge$  TSI  $\wedge$  SI
In2, Out2,In3,Out3  $\in$  TSI
In4, Out4,In5,Out5  $\in$  TSR
Se,Se1,Link,Link1  $\in$  UNS
#( In2 ) < 3  $\wedge$  #( In2 ) > 8
#( In3 ) < = 5  $\wedge$  #( In3 ) > = 10
#( In4 ) < 3  $\wedge$  #( In4 ) > 8
#( In5 ) < = 5  $\wedge$  #( In5 ) > = 10
SR to SI' = SR to SI
UNS to SR' = UNS. Se  $\wedge$  TSR. Out5  $\wedge$  UNS. Link  $\wedge$  TSR. In4  $\wedge$ 
TSR. Out4  $\wedge$  SR. Con1  $\wedge$  TSR. In5  $\wedge$  SR. Req1
NUS to SI' = UNS. Link  $\wedge$  TSI. In2  $\wedge$  UNS. Se1  $\wedge$  TSI. Out3  $\wedge$ 
TSI. Out2  $\wedge$  SI. Req1  $\wedge$  TSi. In3  $\wedge$  SI. Com1
Con1,In2,In3,In4,In5,Rep1  $\in$  receive
Req1,Out2,Out3,Out4,Out5,Com1  $\in$  send

```

从这个例子可以看出，扩展的 Z 语言可以较好地描述动态软件体系。

4 结束语

本文扩展的 Z 语言是对动态软件体系的描述，可以描述软件的动态演化性。笔者拟在以后的研究中通过对动态体系结构的具体环境来选择体系结构语言进行描述，从而达到体系结构动态演化的正确性。由于扩展的 Z 语言是以 Z 语言完全形式化描述为基础，从而使其在将来的研究中能够更好地描述系统动态性。

参考文献：

- [1] BREIVOLD H P,CRNKOVIC I,LARSSON M. A systematic review of dynamic software architecture evolution research [J]. *Information and Software Technology*,2012,54(1):16-40.
- [2] XU Hong-zhen, ZENG Guo-sun. Specification and verification of dynamic evolution of software architecture [J]. *Journal of Systems Architecture*,2010,56(10):523-533.
- [3] LI Jun-cao,PILKINGTON N T,XIE Fei, *et al.* Embedded architecture description language [J]. *Journal of Systems and Software*,2010,83(2):235-252.
- [4] RYOO J,SAIEDIAN H. AVDL:a highly adaptable architecture view description language [J]. *Journal of Systems and Software*,2006,79(8):1180-1206.
- [5] RADEMAKER A,BRAGA C,SZTAJNBERG A. A rewriting semantics for a software architecture description language[J]. *Electronic Notes in Theoretical Computer Science*,2005,130(12):345-377.
- [6] 李千目,许满武,张宏,等. 软件体系结构设计 [M]. 北京:清华大学出版社,2008.
- [7] 余萍,马晓星,吕建,等. 一种面向动态软件体系结构的在线演化方法 [J]. *软件学报*,2006,17(6):1360-1371.
- [8] 王志坚,费玉奎,姜渊清,等. 软件构件技术及其应用 [M]. 北京:科学出版社,2006.